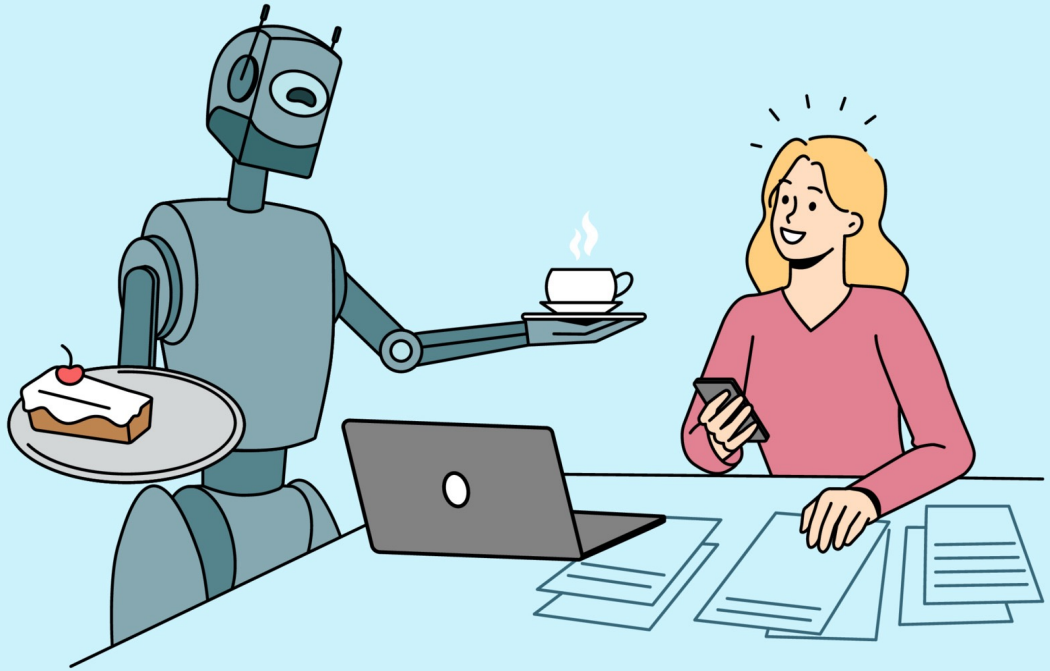
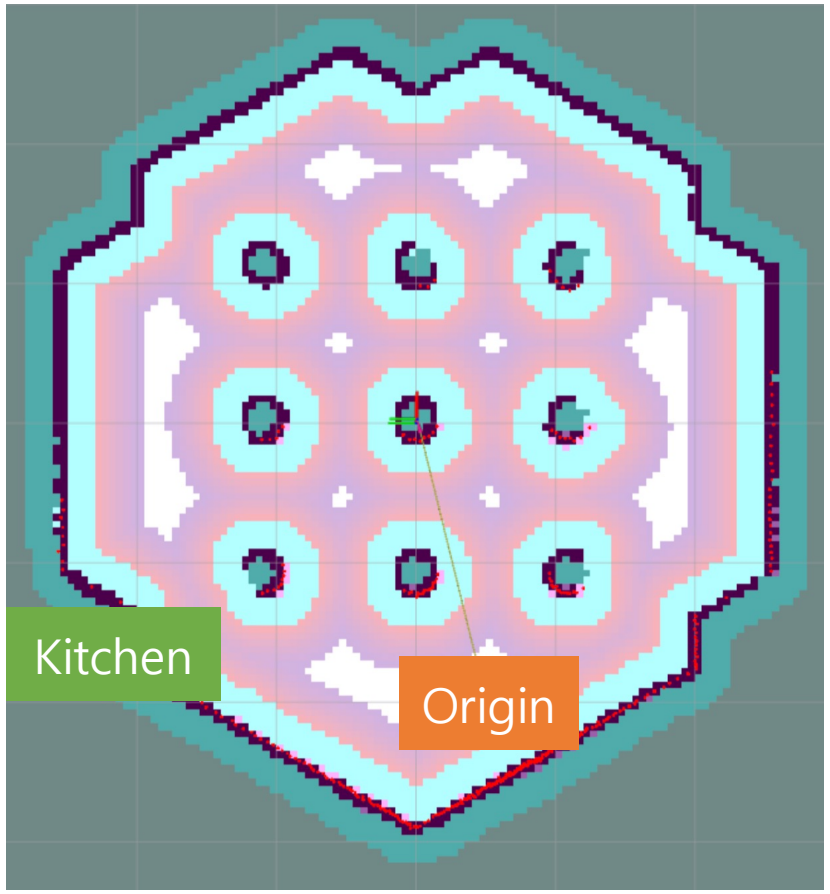


서빙 로봇 시스템 설계 및 구현

목차

컨셉 소개



< 맵 구조 >

배식

- Origin[1] → Kitchen → Customer → Origin[1]

퇴식

- Origin[1] → Customer → Kitchen → Origin[1]

1) Origin : 원점, 로봇이 대기하는 공간

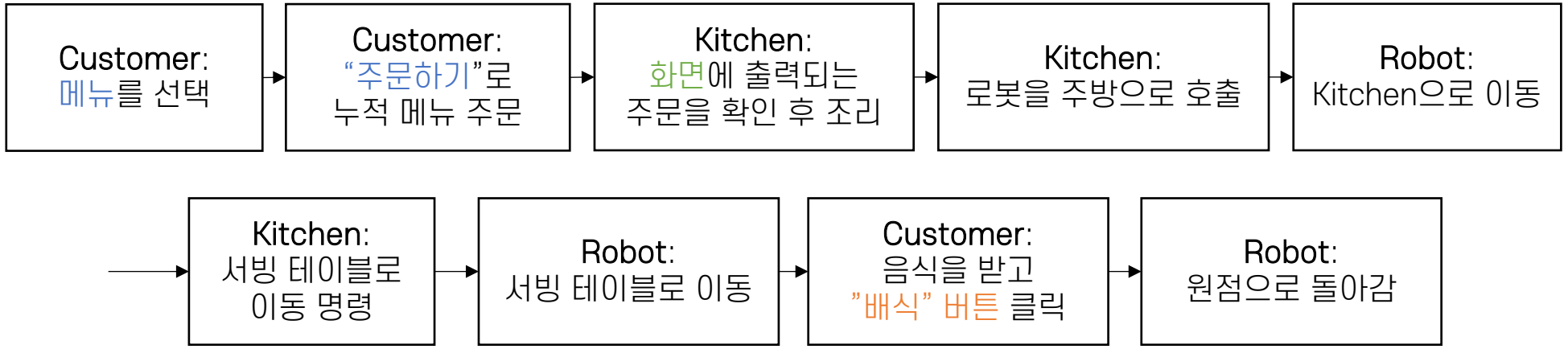
시나리오

Customer UI*

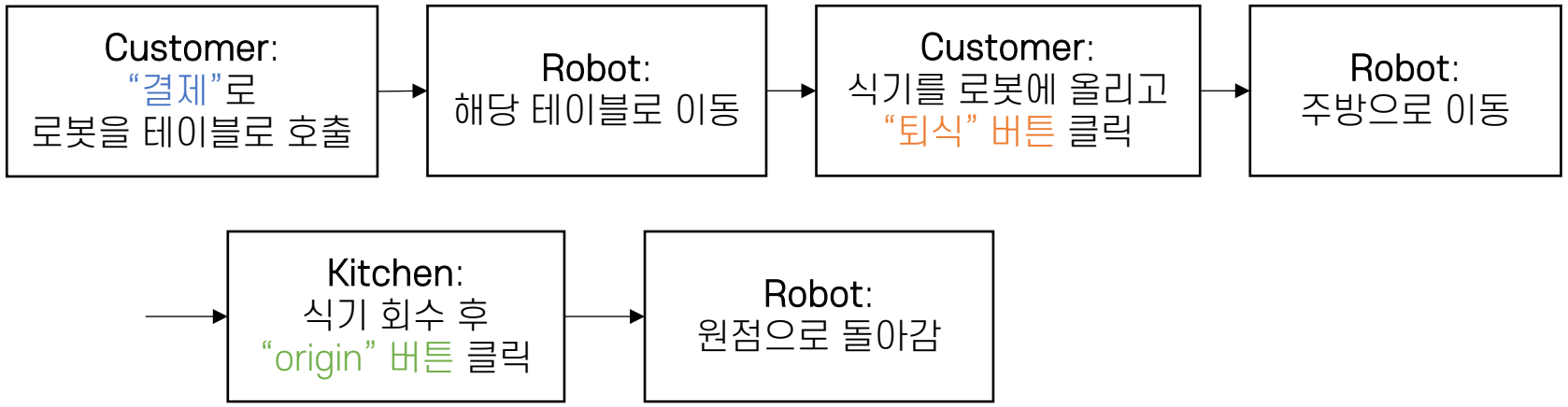
Kitchen UI*

Robot UI*

배식

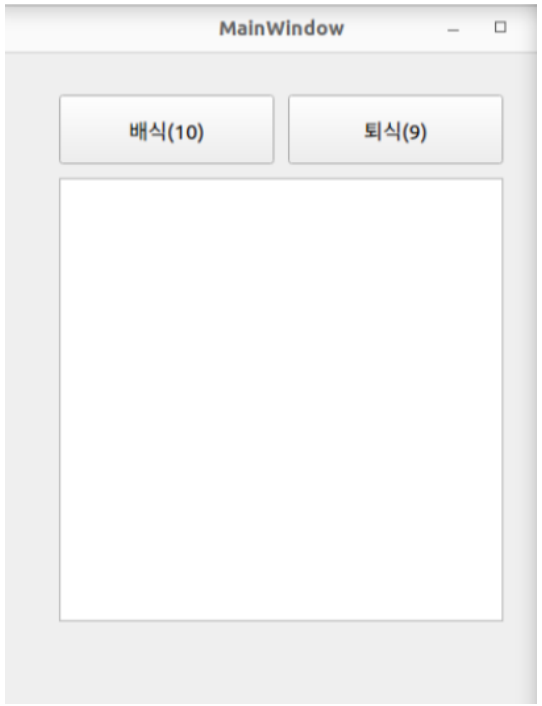


퇴식



화면 UI

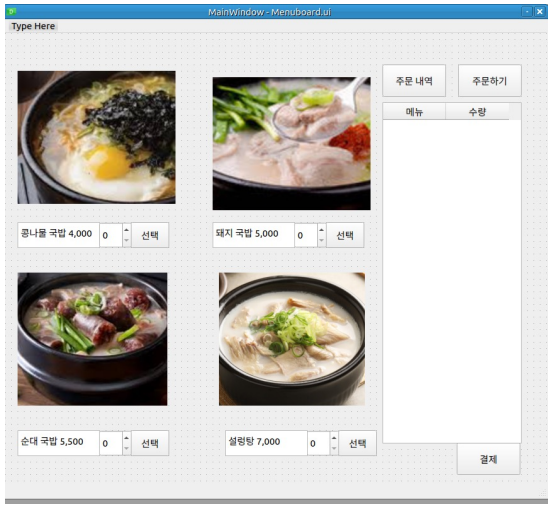
Robot UI



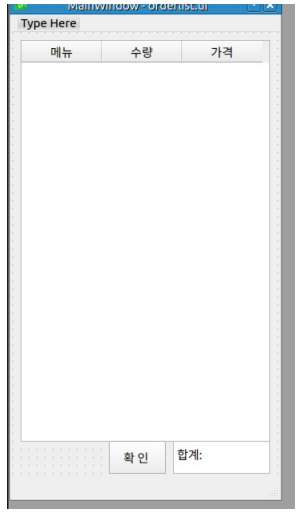
< UI >

- 고객을 위한 화면
- 배식 → 원점 / 퇴식 → 주방

Customer UI

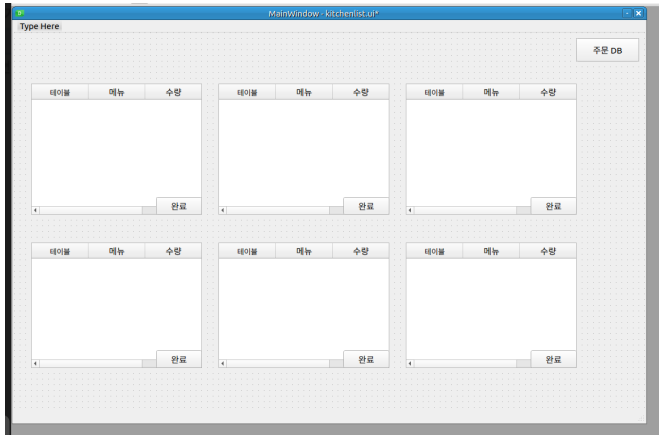


< 주문 화면 >

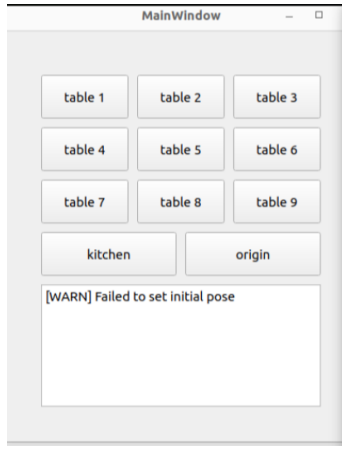


< 주문 내역 확인 화면 >

Kitchen UI

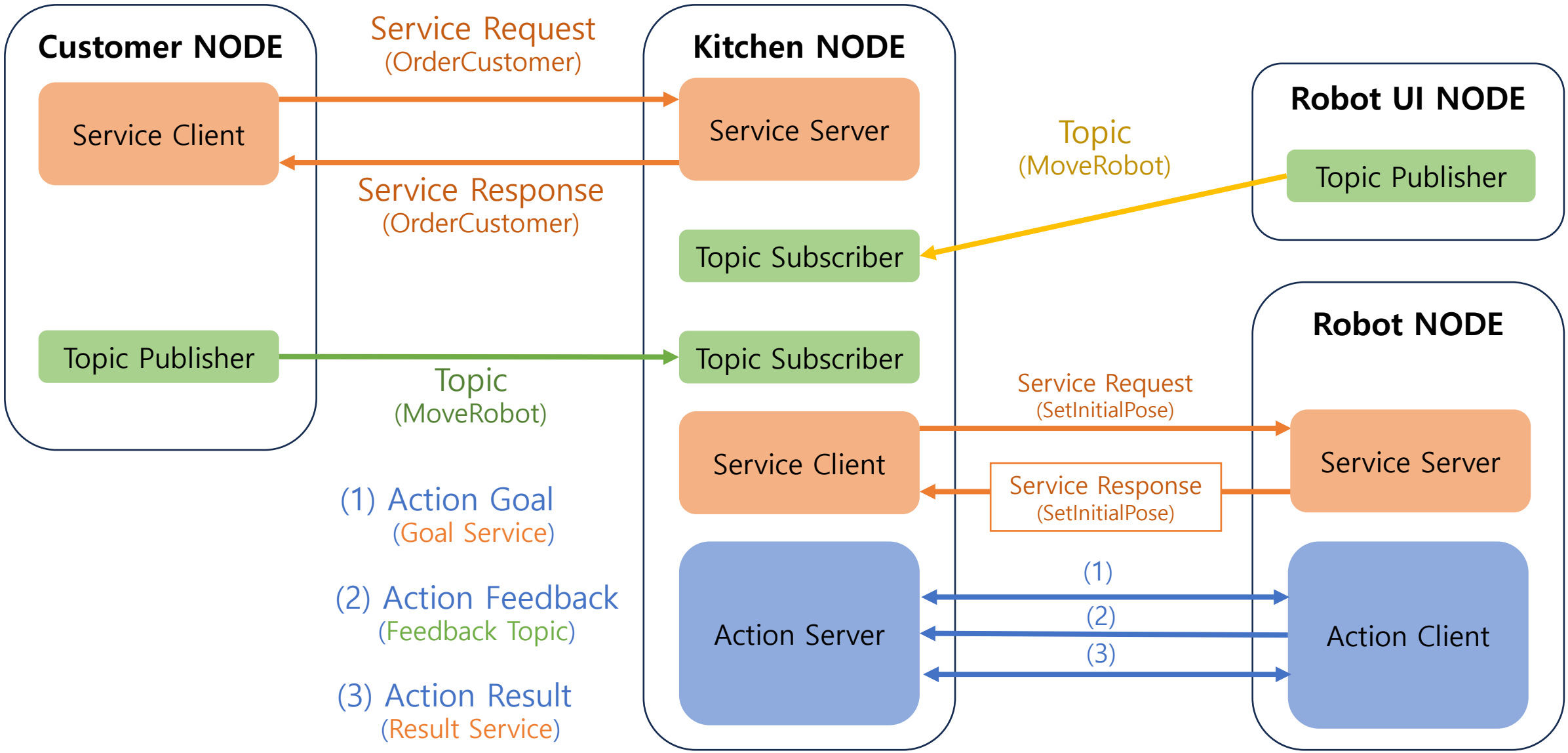


< 주문 확인 화면 >



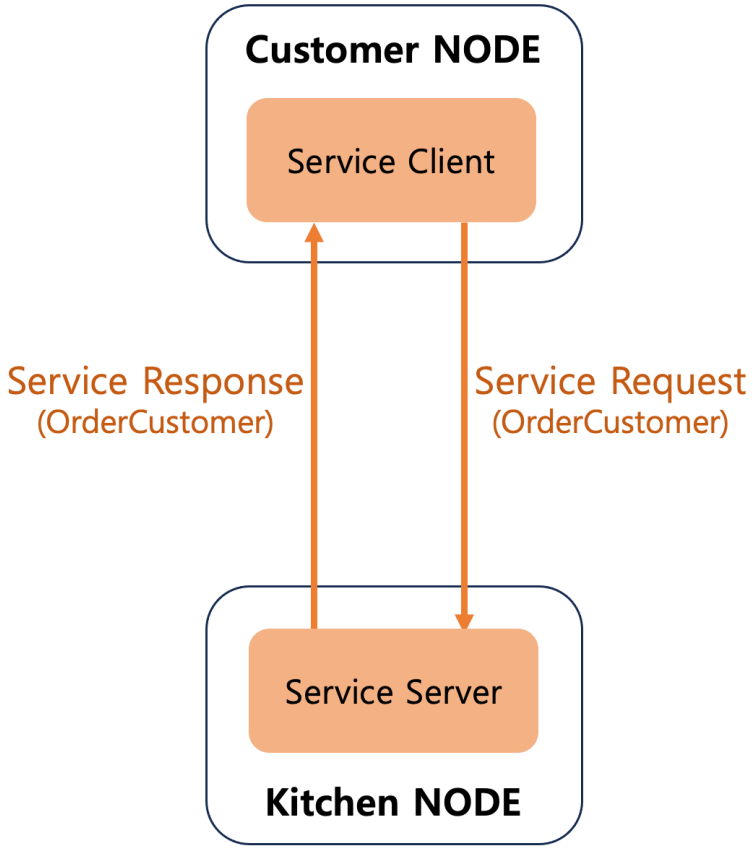
< 로봇 제어 화면 >

Node Graph


Customer NODE
Service Client
Topic Publisher
Service Request
(OrderCustomer)
Service Response
(OrderCustomer)
Topic
(MoveRobot)
Kitchen NODE
Service Server
Topic Subscriber
Topic Subscriber
Service Client
Action Server
Robot UI NODE
Topic Publisher
Topic
(MoveRobot)
Robot NODE
Service Request
(SetInitialPose)
Service Response
(SetInitialPose)
Service Server
(1)
(2)
(3)
Action Client
(1) Action Goal
(Goal Service)
(2) Action Feedback
(Feedback Topic)
(3) Action Result
(Result Service)

Node Graph

배식 - service



Service Request

- 메뉴, 메뉴 수량, 테이블 번호, 주문 차수
- 로깅 레벨 : warning

```
while not self.order_service_client.wait_for_service(timeout_sec=0.1):
    self.get_logger().warning('service not available.')
```

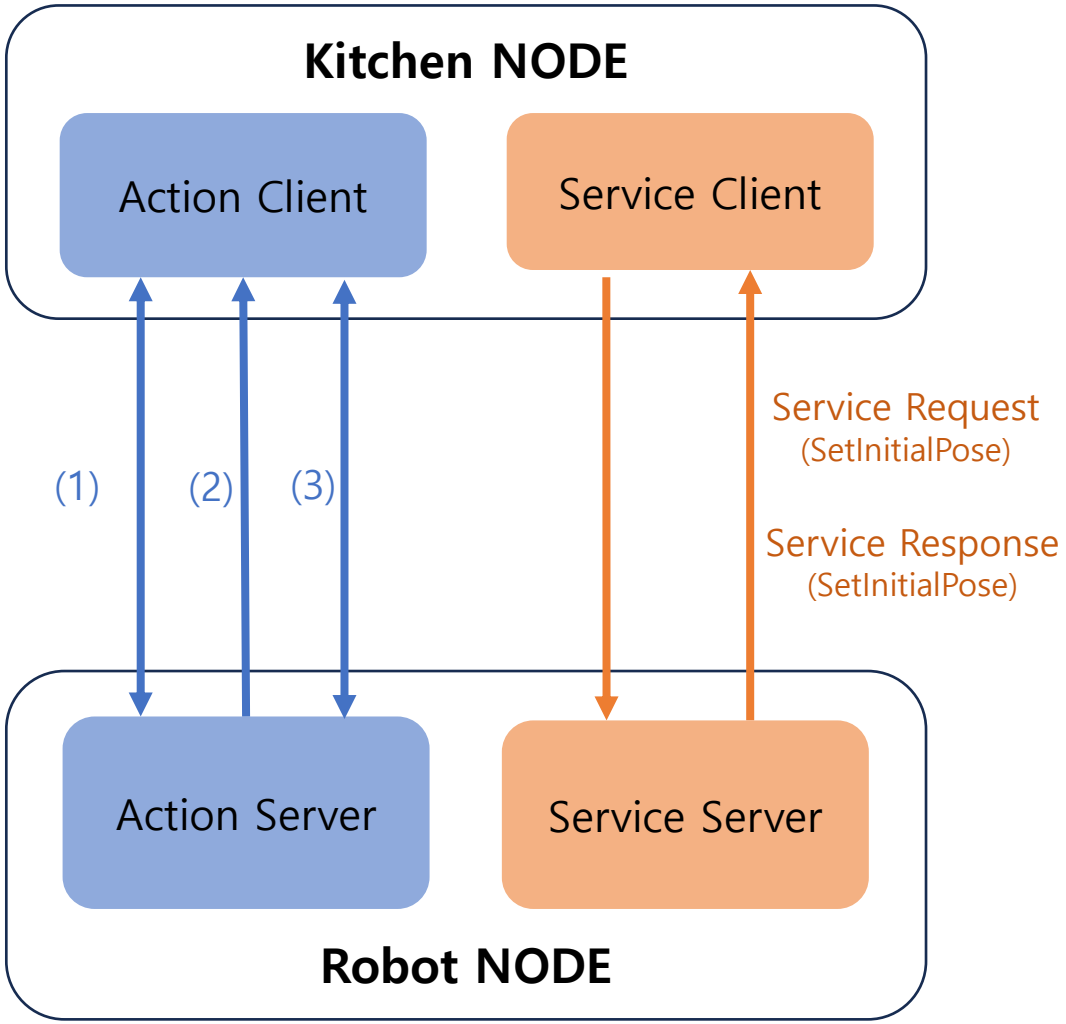
Service Response

- 주문 접수 여부
- 로깅 레벨 : info & error

```
if future.done():
    try:
        response = future.result()
        order_check = 0
        self.customer.get_logger().info(f"Response received: {response}")
        order_check = response.order_check
        if order_check == 1 :
            self.customer.get_logger().info('주문이 정상적으로 접수되었습니다.')
            self.customer.order_round += 1
        else:
            self.customer.get_logger().error('주문 접수 오류 발생!')
    except Exception as e:
        self.customer.get_logger().error(f'Service call failed: {str(e)}')
```

Node Graph

배식 - Action



Action

1) Action Goal: PoseStamped [입력 값]
2) Action Feedback: FeedbackMessage
3) Action Result: ResultMessage

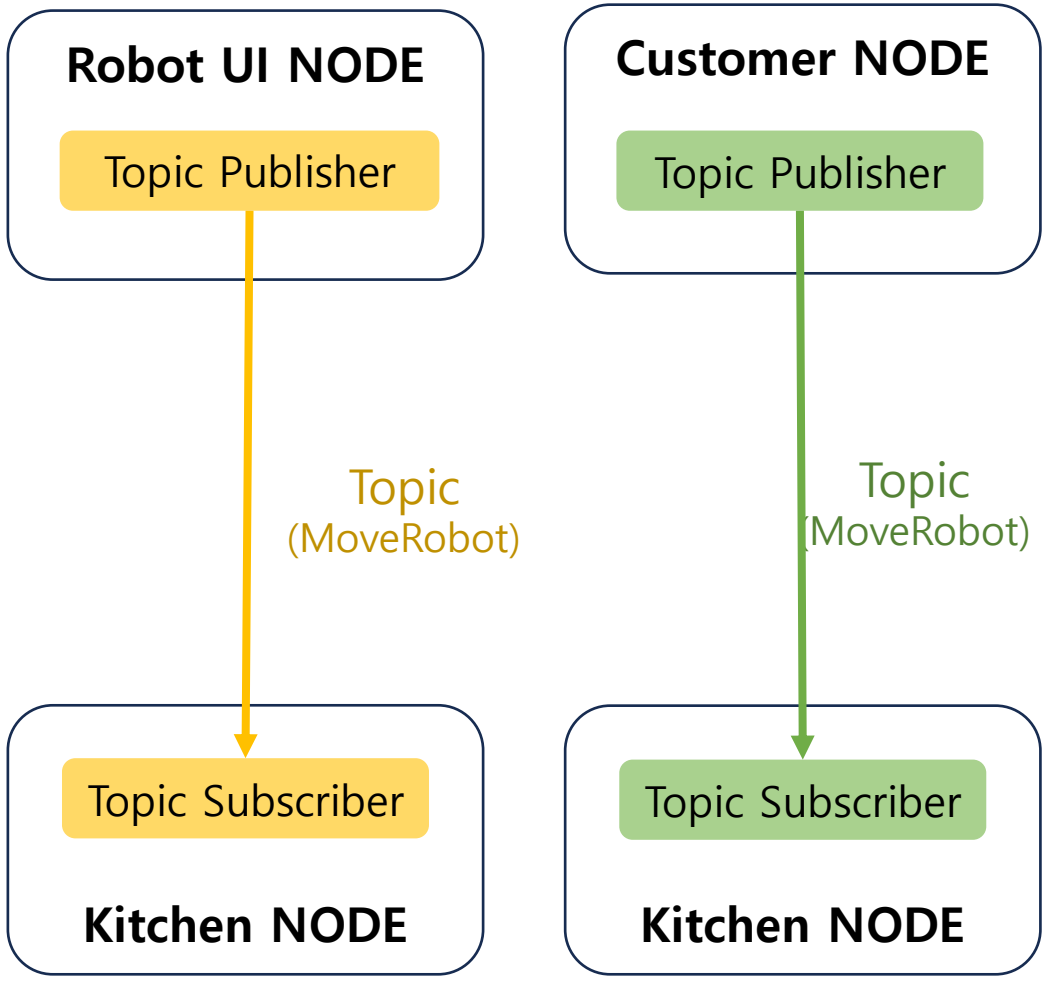
Service

1) Request: 위치, 방향(Quaaternion), 공분산(위치 추정의 불확실성)
2) Response: 초기 위치 설정 성공 여부

```
if future.result() is not None:
    self.get_logger().info("Initial pose set successfully.")
else:
    self.get_logger().error("Failed to set initial pose.")
```

Node Graph

배식 - Action



Topic

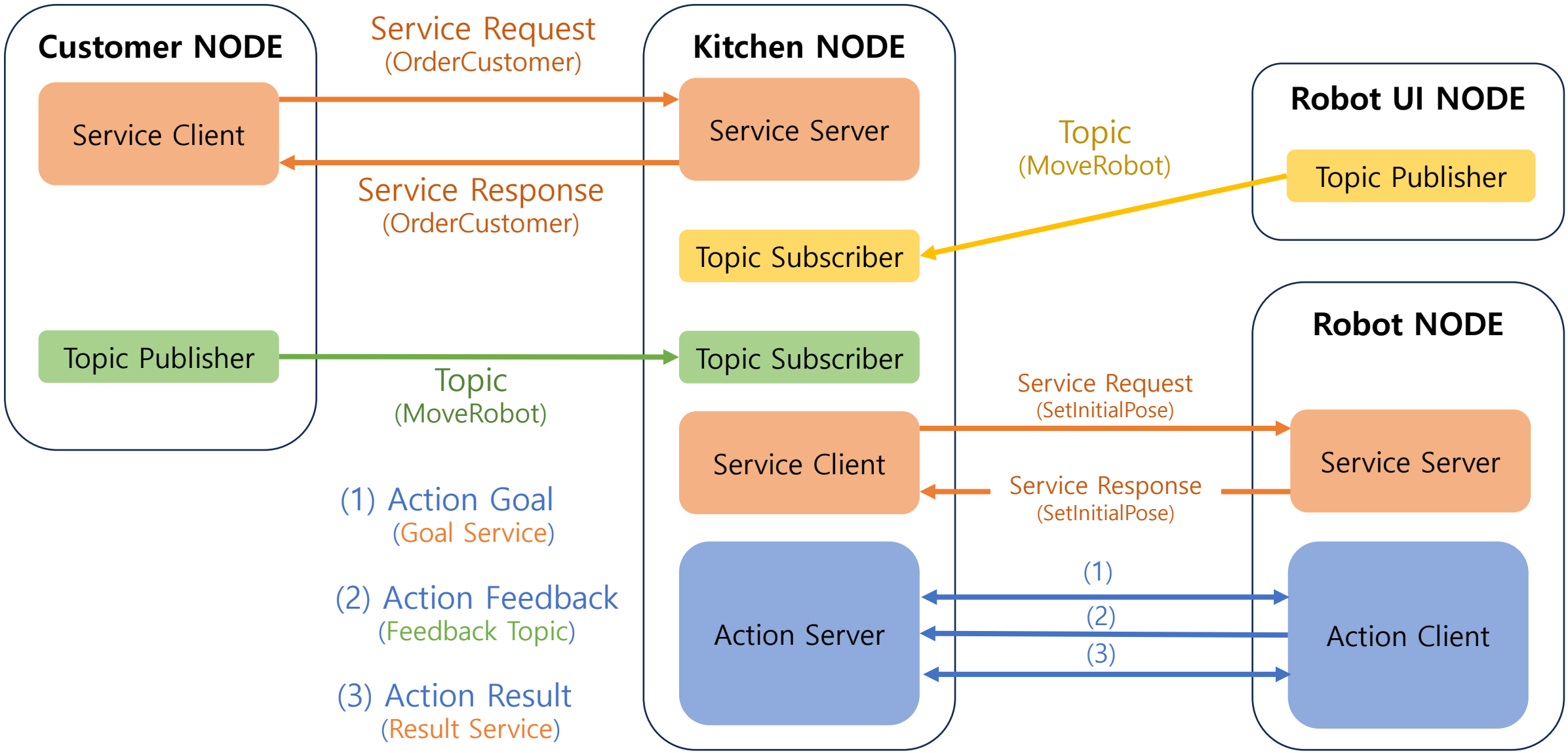
- 로봇이 가야 할 대상 테이블 번호

QoS 설정

- 신뢰성: 모든 메시지가 전달될 때까지 재시도
- 기록 정책: 마지막 N개의 메시지만 유지
- 큐의 깊이 (여기선 'qos_depth' 매개변수 사용)
- 지속성: 발행자가 없어도 메시지를 유지

```
QOS_RKL10V = QoSProfile(
    reliability=QoSReliabilityPolicy.RELIABLE,
    history=QoSHistoryPolicy.KEEP_LAST,
    depth=qos_depth,
    durability=QoSDurabilityPolicy.TRANSIENT_LOCAL
)
```

Node Graph


Customer NODE
Service Client
Topic Publisher
Service Request
(OrderCustomer)
Service Response
(OrderCustomer)
Topic
(MoveRobot)
Kitchen NODE
Service Server
Topic Subscriber
Topic Subscriber
Service Client
Action Server
Robot UI NODE
Topic Publisher
Topic
(MoveRobot)
Robot NODE
Service Request
(SetInitialPose)
Service Response
(SetInitialPose)
Service Server
(1)
(2)
(3)
Action Client
(1) Action Goal
(Goal Service)
(2) Action Feedback
(Feedback Topic)
(3) Action Result
(Result Service)

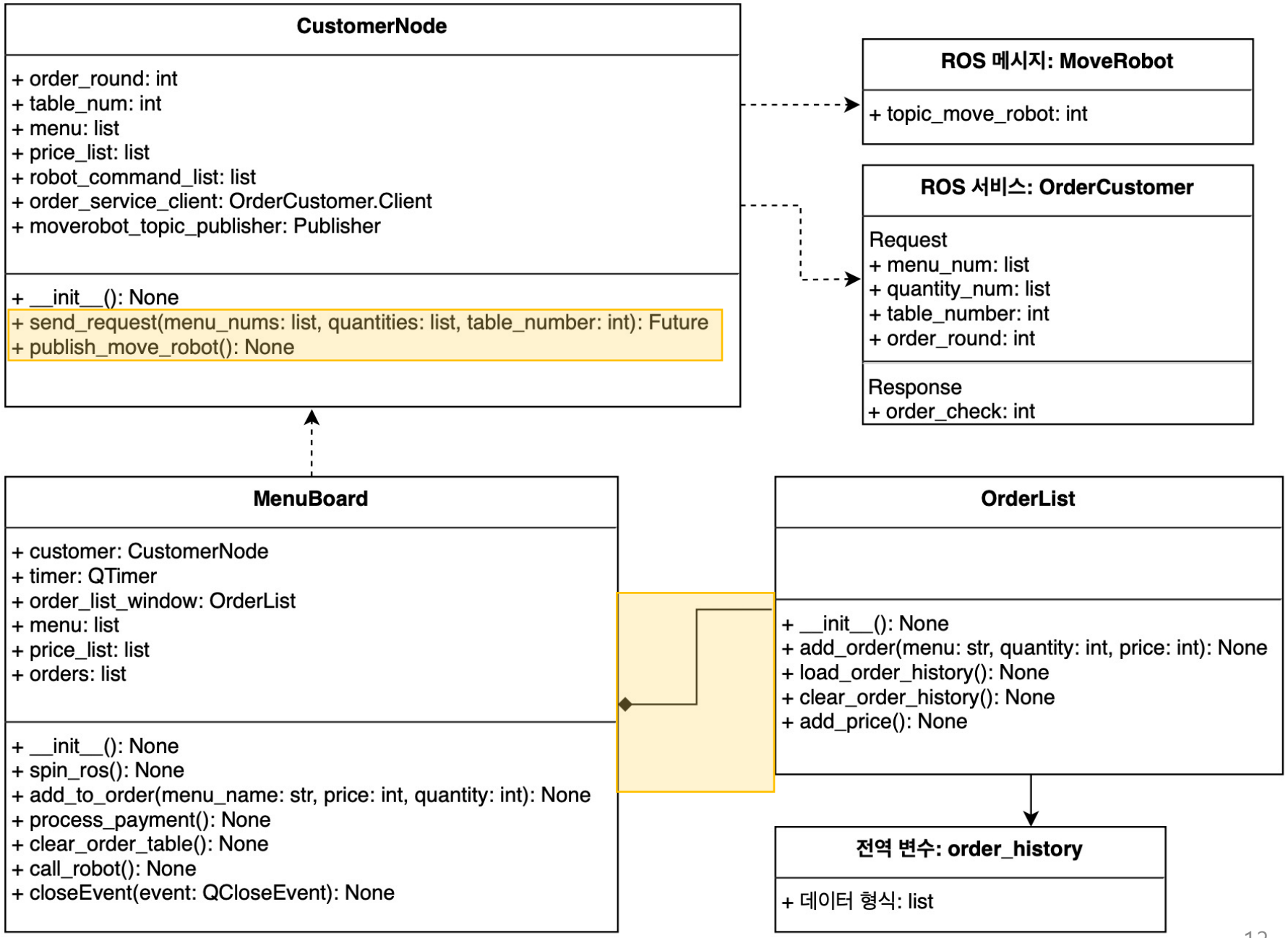
Data Base

customer_order

	Column Name	Type
PK	uuid (고유 ID)	TEXT
	datetime (주문 시간)	TEXT
	menu_name (메뉴 이름)	TEXT
	quantity (수량)	INTEGER
	total_price (주문 가격)	INTEGER

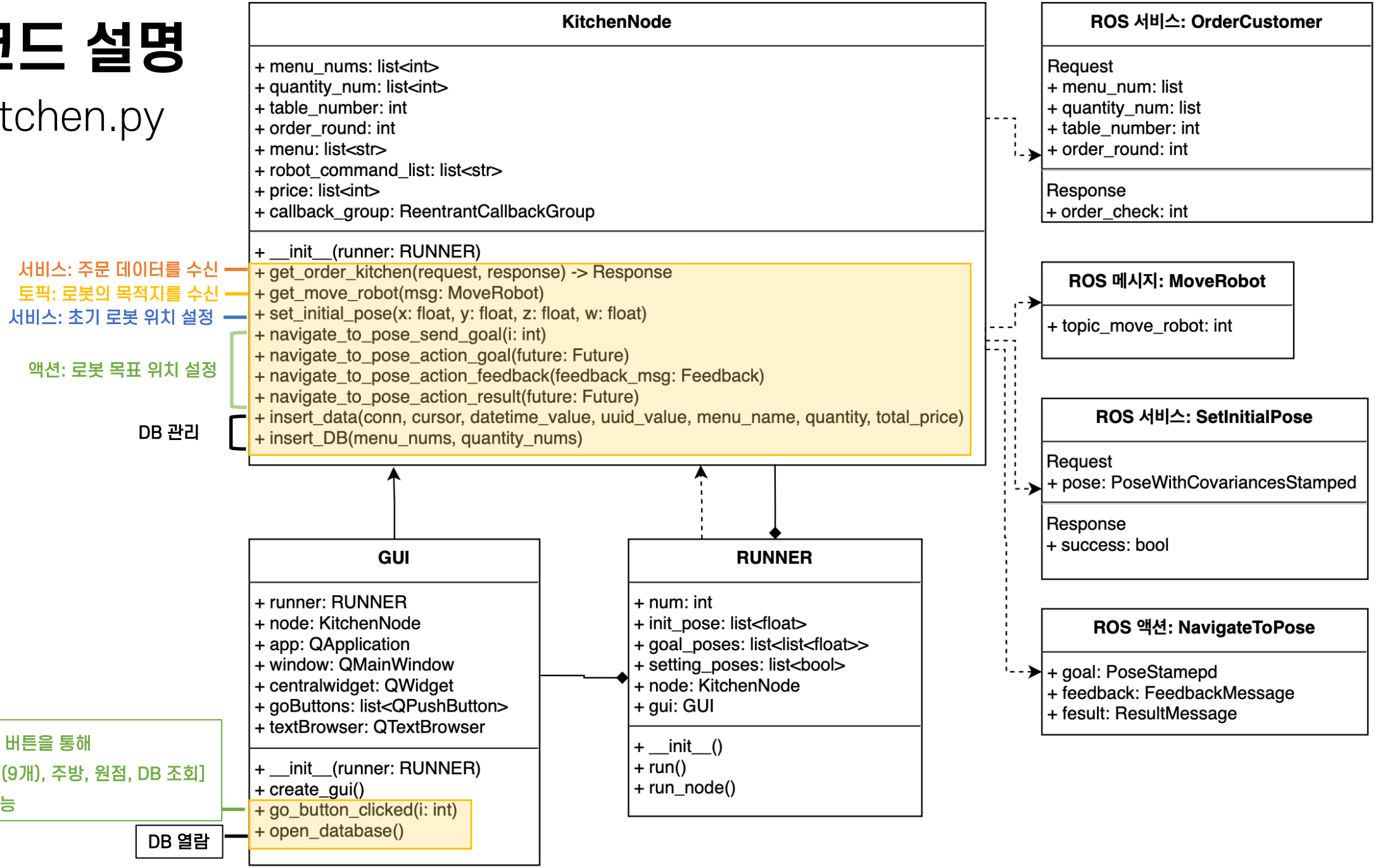
코드 설명

Customer.py



코드 설명

Kitchen.py



코드 설명

RobotUI.py